CLUBE

Setembro/85

N.º 36

AVENIDA DA BOAVISTA, 280-6.º • TELEFONE 65127 • 4000 PORTO

NESTE NÚMERO

Edição: Clube Z80

Fotocomposição: Fotomecânica Mabreu

Impressão: Ramos dos Santos & Ca,. Lda. / Porto

Tiragem: 500 exemplares, Setembro 1985

EDITORIAL

Como se deve ter apercebido, um clube de utilizadores de microcomputadores, não será outra coisa que um grupo de indivíduos que comunica as suas experiências, que pede auxílio aos outros entusiastas desde que em dificuldade, ou que propõe novas ideias ou novos caminhos até para soluções mais económicas.
O modo mais eficaz que encontramos para comunicar uns com os outros, foi através de uma publicação mensal, que circula sem publicidade, e que obviamente todos pagamos os custos da sua impressão e transportes.
Durante três anos, fizemos chegar até cada um dos que acreditam que esta publicação fosse útil, cerca de 20 páginas impressas, todos os meses.
Simplesmente o número de utilizadores que renovavam a taxa de adesão, ia sendo reduzido ou o número daqueles que chegavam de novo não compensava o volume dos que deixavam o nosso grupo.
Temos uma certa tristeza em parar o trabalho que o Fernando Preces tem vindo a produzir, à volta da linguagem máquina e que foi uma das coisas mais interessantes que o Clube Z80 gerou.

Talvez que o grande culpado da paragem desta publicação seja a dificuldade que as pessoas têm em entender que era necessário desde sempre que fossem publicados trabalhos originais ou descobertas e adaptações com que cada um poderia ir enriquecendo o Clube Z80.
Chegamos ao fim meus amigos, mas pode ser que tenhamos terminado no momento exacto. As razões são económicas — os custos já ultrapassaram as receitas — mas também observamos que não existe muita dinâmica por parte dos sócios e isso também ajuda a parar com a publicação.
Para aqueles que eventualmente ainda possuam um valor residual de assinatura, de uma ordem superior a 3 meses, solicitamos que nos peçam cassetes ou livros, que têm sido divulgados ao longo dos 37 números do Clube Z80. Procuraremos compensar os valores das assinaturas de uma forma que nos pareça razoável.
Gostamos de ter criado o Clube Z80, com sinceridade e sentimo-nos satisfeitos com o sentimento do «dever cumprido».

Um grande abraço
Alexandre Sousa

INTRODUÇÃO AO CÓDIGO MÁQUINA

Autor: FERNANDO PRECES
SACAVÉM

(Cont. dos números anteriores)

GRUPO 17 — As instruções de interrupção

No Assembler do Z80, existem 7 instruções que permitem manipular, através de três modos de actuação, o seu sistema de **interrupts.**

Mnemónicas	Códigos	Comentários
IM0	237,70	MODO 0
IM1	237,86	MODO 1
IM2	237,94	MODO 2
DI	243	incapacitar interrupções
EI	251	permitir interrupções
RETI	237,77	retorno de interrupção não prioritária
RETN	237,69	retorno de interrupção prioritária

Utlizando duas linhas de controlo para interrupts, o CPU suspende a sequência normal da programação em curso, para efectuar uma leitura do teclado, produzir novos sinais para a vídeo, ou saltar para determinadas rotinas de interrupção, impostas por programa impresso em memória, ou por imposição de qualquer outro dispositivo.
Quando um interrupt é aceite, o conteúdo do registro PC (e contador de programa) é colocado no Stack, salvaguardando o retorno ao programa interrompido. PC é então carregado com o endereço da rotina de interrupção imposta, iniciando o CPU a nova tarefa até surgir uma instrução que desactive o interrupt, promovendo o retorno. Mais uma vez o registro PC é carregado, agora com o endereço depositado no Stack, voltando e CPU é tarefa abandonada.
A primeira linha de controlo, chamada NMI — Non maskable interrupt (interrupção não mascarável ou prioritária), **obriga** o CPU a saltar, não abruptamente mas após a execução de qualquer instrução em curso, para o endereço ROM 102 — Non maskable interrupt rotine, independentemente do MODO em que esteja programado.
Na ROM do ZX81, esta rotina encaminha a produção de um novo Display cujo endereço de origem se encontra nos conteúdos da variável D. FILE. Terminada a projecção do Ficheiro de Imagem, o interrupt é desactivado e o CPU retorna ao programa interrompido.
No Spectrum, se este tipo de interrupt é activado, a rotina ROM 102 testa os 2 bytes da variável NMI ADD (endereços 23728/9) e provoca um RESET ao Sistema — salto para o endereço 0 da ROM, quando o valor desses bytes for zero.
Voltaremos a falar desta rotina no capítulo dedicado à ROM.
A segunda linha de controlo, chamada INT (linha de interrupção mascarável ou não prioritária) conduz um sinal de interrupt que o CPU pode ignorar, quando previamente programado com a instrução DI — Disable interrupt (impedir interrupções). Se tal impedimento não existir, este interrupt força o CPU a saltar para a Rotina 56 da ROM — The maskable interrupt rotine.
Esta rotina verifica os impulsos contados pela variável FRAMES, incrementa-a se necessário para actualizar o relógio interno do CPU e acciona uma leitura ao teclado.
Quando o Spectrum é ligado, a linha de controlo INT é desactivada assim que o CPU acabar de executar a primeira instrução da ROM. Esta instrução (DI) vai **impedir** qualquer interrupção para que o CPU possa em paz, formatar o sistema seguindo a rotina de iniciação. Na fase final da rotina uma instrução EI activa a linha de controlo e ao surgir no écran a

mensagem C 1982 Sinclair Research Lta o CPU está seleccionado no MODO 1 para interrupts e disponível para o utilizador.

Modos de interrupção. — ZX81 e Spectrum aceitam 3 Modos programáveis para uso do Sistema de interrupts.

MODO 0 — Quando seleccionado neste modo, se um interrupt é activado, o CPU **espera** que um DATA seja introduzido BUS de Dados indicando o endereço duma RST Routine a colocar no contador de programa.
Assim, qualquer dispositivo periférico quer interno (do próprio computador) ou externo, tem acesso a 8 importantes rotinas da ROM.

MODO 1 — Neste modo, quando activado um interrupt, o CPU salta para o endereço 56, uma rotina RST já referenciada em cima.

MODO 2 — A utilização deste modo é um pouco mais complicada, mas a única forma de uma fase de interrupção fazer o CPU executar uma rotina que esteja situada na RAM, o mesmo é dizer **uma rotina construída pelo utilizador**. Vejamos então quais as condições necessárias para se obter uma interrupção com o CPU seleccionado em Modo 2.

a) O CPU terá de receber pelo BUS de Dados um byte enviado pelo dispositivo que requere a interrupção. (Qualquer periférico ligado ao computador).

b) O CPU vai usar o registro I (também chamado Vector de Interrupção), copiando o seu conteúdo para formar, como byte mais significativo, um vector que indique o endereço a seleccionar numa tabela de interrupts. O DATA copiado da BUS de Dados, será o byte menos significativo desse vector.

(Exemplo: copia do registro I * 256 + DATA)

c) O CPU carrega o registro PC com o endereço retirado dessa tabela, que poderá corresponder a uma rotina da ROM ou a qualquer rotina situada da RAM, escrita para o efeito.

Como bem sabemos, o registro I não é um registro de utilização e as suas funções no interior do CPU são muito especializadas. Carregar este registro com valores entre 64 e 127 inclusiveis afecta o trabalho do CPU e compromete a imagem.

Assim, para criar este tipo de interrupt, poderemos carregar o registro I com valores de (0 a 63 e 128 a 255) que nos permitem acesso às seguintes faixas das memórias:

a) 0 a 16383 — Nestes endereços, pertencentes à ROM, não nos é possível alterar o seu conteúdo para escrever uma tabela de interrupts, mas não é difícil (ver programa abaixo) testar essas faixas da memória, e encontrar dois números seguidos (n * 256 + n1) para formar um endereço que aproveite uma zona livre da RAM aonde possamos colocar a nossa rotina de interrupt.

Programa «Endereços para a rotina de interrupt»

```
  5 REM 1.ª FAIXA DE VALORES (REGISTRO I)
 10 FOR N=0 TO 63
 20 REM ENDEREÇO ROM = N * 256 + 255 (255 n.º FIXO
    do BUS DE DADOS — Spectrum sem periféricos)
 30 LET E = N * 256 + 255
 40 REM ENDEREÇO RAM obtido a partir de 2 números da
    ROM
 50 LET RAM = PEEK E +256 * PEEK (E+1)
 60 REM ESCRITA DOS ENDEREÇOS RAM
 70 PRINT RAM
 80 NEXT N

100 REM 2.ª FAIXA DE VALORES (REGISTRO I)
110 FOR N = 128 TO 255
120 REM ENDEREÇOS RAM (TABELA DE INTERRUPTS)
130 LET ER = N * 256 + 255
140 PRINT ER
150 NEXT N
```

b) 32768 a 65535 (Parte 2 do programa acima) — com a 2.ª faixa de valores para o registro I, podemos escolher a zona livre da RAM aonde colocar a tabela do endereço da rotina de interrupção.

Apresentarei de seguida um exemplo para activar a tecla BREAK num programa em Código máquina.
Esta rotina será utilíssima, quando abrir um programa para o alterar, ou ainda para colocar em programas a ensaiar, nos pontos aonde possam surgir problemas.
Se não tiver periféricos ligados ao computador, quando seleccionar o MODO 2 de interrupts, o DATA que surge no respectivo BUS para formar o endereço da tabela é o byte 255.
Dois números da ROM vão servir para formar o endereço da Rotina que ficará instalada no topo da RAM.

Para carregar o registro I, foi escolhido o byte 9.

9 * 256 + 255 = 2559
O endereço 2559 contém o byte 105
O endereço 2560 contém o byte 254

254 * 256 + 105 = 65129 (início da rotina BREAK)

	Mnemónicas	Comentários
	ORG 40000	Rotina em ensaio
LOOP:	DI	Impede interrupções
	LD BC 20	
	LD HL 1040	
	ADC HL BC	
	LD (45000) HL	
	etc., etc.	Ponto duvidoso (por exemplo)
	LD A 9	Arranque do interrupt
	LD I A	Carga do registro I
	IM 2	Selecção do MODO. CPU memoriza endereço de retorno, e salta para a rotina 65129
	LD A 62	Retorno do interrupt
	IM 1	Normalizar as condições de trabalho
	LD I A	
	JR LOOP	Salta para o início
	ORG 65129	Rotina BREAK
	DI	Protecção dos conteúdos dos registros
	PUSH AF	
	PUSH BC	
	PUSH DE	
	PUSH HL	
	PUSH IX	

```
          LD A 127     }  Teste da tecla BREAK
          IN A (254)   }
          RRA             Rotação
          JR NC BREAK     Salta se Carry = 0
          POP IX
          POP HL          Recupera-se o conteúdo
          POP DE          dos registros
          POP BC
          POP AF
          EI              Permissão de interrupts
          RET             Retorno à rotina em ensaio
BREAK:    EI              Subrotina BREAK
          LD A 62
          IM 1
          LD I A
          RST 8           ROM — Rotina de erro.
          DEFb 20         Emite mensagem de erro
                          BREAK E retorna ao Basic
```

Instrução DI — Esta instrução força o CPU a ignorar interrupções. Um flag chamado IFF 1 (1.° Flips — flops de interrupção) é colocado a zero quando a instrução é executada, incapacitando o pedido de interrupts não prioritários (Exemplo: o CPU não atende o teclado). O conteúdo dos registros não é afectado por esta instrução.

Instrução E1 — esta instrução autoriza o CPU a aceitar interrupts. No entanto, a sua validade só é reconhecida após a **execução da instrução seguinte.** O Flag IFF 1 é colocado a 1.

Instrução RETI — Efectua o retorno dum interrupt e envia para o exterior um sinal de disponibilidade. (Não utilizada no Spectrum).

Instrução RETN — Efectua o retorno dum interrupt prioritário. (também não é utilizada).

GRUPO 18 — Foram aqui agrupadas as últimas instruções do Assembler Z80. São instruções independentes e com funções diversas.

Mnemónicas	Códigos
CPL	47
NEG	237,68
SCF	55
CCF	63
HALT	118
DAA	39

CPL — Esta instrução permite complementar o registro A. A operacão consiste em mudar os 8 bits para o seu estado oposto o que aritmeticamente corresponde à subtracção 255 — Valor de A. Os flags não são afectados.

NEG — Esta instrução permite a formacão do 2.° complemento aritmético no registro A. Representa a seguinte operação aritmética:

256 — Valor de A

Afecta todos os flags. O flag Sinal reflecte a polaridade do resultado. O flag Carry dá **um** se o resultado de operação for **zero**, o flag O/P dá **um** se o resultado for superior a 127 e o flag zero dá um se o resultado der zero.

HALT — Imediatamente após a execução desta instrução o CPU suspende a sequência do programa, parando as suas funções para o exterior, e executar instruções NOP até receber instruções da próxima interrupção, saltando então para a rotina 56 da ROM que executa e daí retorna do programa que seguia.

Atenção — Não deve executar uma instrução HALT sem primeiro activar a permissão de interrupt com uma instrução EI, caso o programa se encontre sob a acção duma instrução DI, porque a flag IFF 1 tem de ter o valor (1) para que a instrução HALT não provoque um **Crash.**

No Spectrum o tempo de paragem do CPU cm uma instrução HALT terá mais ou menos a mesma duração que em BASIC terá a PAUSE 1.

No ZX81 a instrução HALT tem o mesmo código dum NEW LINE. Não é por coincidência.

Enquanto o circuito de vídeo transfere uma linha de imagem, a última instrução lida pelo CPU é um NEW LINE e que este vai reconhecer como uma instrução HALT. O CPU **trava,** aguarda pela interrupção para a projecção dessa linha de imagem e retorna novamente a sua tarefa. No modo SLAW a instrução HALT também serve de compasso de espera para que se obtenham determinadas paragens necessárias a este modo.

SCF — Esta instrução ajusta o flag Carry no valor 1. Os outros flags não alteram.

Exemplo: Carga (LOAD) dum programa sem cabeça.

ORG Zona disponível da RAM fora da extensão do programa a carregar.

```
INÍCIO:   LD A N       (N) o número que define o bloco
          LD IX        (1.° byte do programa a carregar)
          LD DE        (comprimento em bytes do
                       programa)
          SCF          SET Carry flag
          CALL 1366    rotina ROM (carga de bytes)
          RET
```

NOTA: Tipo de bloco — Imediatamente a seguir às faixas vermelhas (a cor é dependente da velocidade de transmissão) o primeiro byte a surgir indica o tipo de bloco. Se o valor introduzido no registro A não coincidir o programa não **carrega.**

CCF — Complemento do Carry Flag. Esta instrução **inverte** o estado deste flag.

DAA — Decimal aritmético em código Binário. Esta instrução coloca o registro A a trabalhar neste sistema aritmético. (Sem interesse actual).

CAPÍTULO 3 — 16 k ROM, o monitor do SPECTRUM

Neste texto já por várias vezes me referi à ROM do Spectrum. A primeira, falando das suas características e descrevendo o seu funcionamento, nas restantes, fazendo referência, em programação de ensaio, de como utilizar algumas das suas rotinas. Porquê dedicar-lhe agora um capítulo inteiro?

Aos entusiastas do Código máquina, poderei apontar 3 importantes motivações para nos envolvermos num estudo desta natureza.

a) Muitas das rotinas ou subrotinas do programa monitor podem ser chamadas pelo utilizador, para executar inúmeras tarefas, evitando perdas de tempo e ocupação de RAM.

b) O monitor mostra-nos como a Sinclair resolveu

certos problemas e as técnicas utilizadas, podem ser copiadas no desenvolvimento das nossas próprias rotinas.

c) O monitor é um programa em código máquina de grande extensão e é instructivo observar como um programa de tais dimensões pode ser estruturado.

Aprender a servirmo-nos o mais possível das rotinas ROM, será o nosso principal objectivo. Para as usarmos, temos no entanto de saber aonde se encontram, para que servem e de que parâmetros necessitam. Não é tarefa fácil, mas a recompensa é tentadora.

Iniciamos a primeira parte do nosso estudo dividindo a ROM em 3 grandes blocos e fazendo um *resumo* do seu conteúdo.

1) As Rotinas do SISTEMA OPERATIVO
2) As Rotinas do INTERPRETADOR BASIC
3) As TÁBUAS DE CARACTERES

BLOCO 1 — Sistema operativo

ROTINA DE INICIAÇÃO — Está localizada nos endereços 0 a 7 e a continuação nos endereços 4555 a 4769.

Uma das principais funções desta rotina é testar a memória e catalogar as variáveis do Sistema, depois de lhes atribuir os valores apropriados.

Esta rotina finda com o aparecimento da mensagem «Sinclair Copyright», saltando o CPU para a primeira Rotina de Execução.

ROTINA PRINCIPAL DE EXECUÇÃO — Está localizada nos endereços 4770 a 5501.

No Spectrum, esta rotina pode ser considerada, tal como o nome indica, a rotina dominante do programa monitor. A partir dela, para que uma linha de Basic seja adicionada à area de programa, 3 rotinas podem ser chamadas.

a) a rotina de comando de LISTAGEM
b) a rotina EDITORA
c) a rotina de TESTE DE SINTAXE

Um comando directo, sem número de linha, após o teste de Sintaxe, é executado de imediato pela chamada da rotina LINE-RUN. Um relatório retirado da Tabua das Mensagens de Erro (localizada nos endereços 5009 a 5430) é projectado na parte inferior do écran, e o CPU volta a saltar para a rotina Principal de Execução, aonde *permanece* aguardando instruções.

ROTINA EDITORA — Está localizada nos endereços 3884 a 4263. Esta rotina, permite a construção duma linha Basic na parte inferior do écran. Na realidade, a linha é formada numa zona da RAM chamada área Editora e copiada a cada passo para o écran.

Quando ENTER é premido e após o teste de SINTAXE, outras rotinas vão ser chamadas para copiarem a linha recém-formada, enviando-a para a área do programa Basic, com a sua posição listada no écran. Segue-se o retorno à rotina Principal de Execução.

Sempre que uma tecla é premida (com excepção das teclas de execução directa, exemplo: ENTER-NEW-CLS, etc.), um salto é formado para a Rotina Editora.

ROTINA KEYBOARD — INPUT — Está localizada nos endereços 4264 a 4380. Ela é chamada através da Rotina Editora e o Salto sempre vectoriado pela área do Canal de informação. Retorna com o código da última tecla premida, obtido pela leitura da variável do sistema, LAST-K.

Outras operações são executadas no interior desta rotina tais como o acerto do flag CAPS LOCK, o câmbio do MODO de operação e ainda o restauro do Segundo bit do control de Cor introduzido pelo teclado, variável do Sistema, K—DATA.

A pesquisa do teclado efectua-se em cada 1/50 do segundo. Esta operação envolve 6 subrotinas que ocupam as localizações 654 a 783.

A principal subrotina de pesquisa de tecla, a KEY-SCAN, localizada de 654 a 702, retorna com o valor da tecla, ainda não descodificado, dentro do registro DE. A conversão desse valor no respectivo Código de Caractere é efectuado pela subrotina KEYBOARD DECODING, a descodificadora.

ROTINA PRINT — OUTPUT — Está localizada nos endereços 2548 a 3405, sendo sem dúvida a mais importante de um grupo de rotinas PRINT. É chamada por intermédio da instrução RST 16 quando o seu endereço for obtido através da área do Canal de Informação.

O caractere a inscrever no écran de TV ou na printer cujo código é transportado no registo A, será entregue pela RST 16 à adequada rotina PRINT, seleccionada pelo tipo de Canal aberto.

A Rotina PRINT OUTPUT, manda imprimir qualquer tipo de caractere (caracteres simples, de control ou de comando). Ela testa repetidamente o flag que determina exactamente *aonde e quando* a saída desse código é autorizada.

Quando o caractere retido se destina ao écran de tv, a posição PRINT é obtida através das respectivas variáveis do sistema, que depois de lidas, são alteradas para a posição seguinte e armazenadas.

Uma posição PRINT, indica o número de linha e coluna da área disponível imediatamente a seguir ao último caractere impresso no écran, bem como o correspondente endereço, na Fila de Projecção, do Pixel esquerdo do Topo (top-left-pixel) da área do caractere.

A rotina PRINT-OUTPUT, comanda as subrotinas que são chamadas a intervir no momento da transferência dos 64 bits, do caractere a enviar para a posição já estabelecida, no Display File (Ficheiro de projecção). Ainda uma outra subrotina será chamada para transferir **o byte atributo,** correpondente ao caractere, para a área de Atributos do Ficheiro, de acordo pelas respectivas variáveis do Sistema.

A rotina PRINT-OUTPUT transfere, não apenas os caracteres que fazem parte da Tábua de caracteres ROM, como também qualquer **caractere definido** que seja criado na área dos UDGs. (Gráficos definidos pelo utilizador).

BLOCO 2 — As rotinas do Interpretador Basic. O interpretador é chamado para testar Sintaxe de uma linha de Basic e a sua respectiva execução, através da rotina LINE-RUN.

A primeira parte da rotina é utilizada em ambas as funções, com o flag RUN-SINTAXE repetidamente testado para determinar se há operações a realizar e quando podem ser executadas.

Vamos considerar o exemplo da seguinte linha de basic:

LET A = 25

A secção Sintaxe vai efectuar o exame **gramatical** do Basic e a secção de Execução de linha (LINE-RUN), terá de atribuir o valor 25 à variável A.

A TABUA DOS COMANDOS — Está localizada nos endereços 6728 a 6934.

O Sistema no Spectrum tem 50 Comandos Basic e a Tábua desses comandos, contém os detalhes de **CLASSE** a que pertence cada um deles, bem como os endereços das respectivas rotinas.

ROTINA CONTROLADORA — Esta é a rotina principal do interpretador de Basic. Encontra-se localizada nos endereços 6935 a 7168 e contém as instruções de Control que permitem a interpretação passo-a-passo de um conjunto de instruções Basic.

À entrada para o teste de Sintaxe é iniciada no endereço 6935 e a operação LINE-RUN no endereço 7050.

ROTINA CLASSE DO COMANDO — Está localizada nos endereços 7169 a 7389 e é responsável pela anlise dos parâmetros que aparecem a seguir a uma instrução de Comando em Basic.

Consideremos 2 exemplos:

O Comando NEXT é definido como comando de Classe 4, por ser seguido por uma variável.

O Comando POKe é definido como Comando de Classe 8, porque necessita de ser seguido de 2 expressões numéricas separadas por uma vírgula.

As rotinas dos Comandos Basic — A maior parte destas rotinas encontram-se localizadas nos endereços 8990 a 9466. As restantes, relacionadas com procedimentos IN-OUT estão localizadas nos endereços destinados ao Sistema Operativo (Bloco 1).

O Salto para qualquer destas 50 rotinas é preparado na rotina principal do Interpretador.

Consideremos o exemplo de 2 linhas Basic:

```
10 CLS
20 GOTO 50
```

No primeiro grupo a ser interpretado, CLS é um Comando sem operando. A respectiva rotina é procurada na Tábua dos comandos e a sua localização obtida no endereço 6846.

Nessa localização da Tábua, encontramos o tipo de Classe atribuído ao Comando (Classe 0) e o respectivo endereço da rotina (3435), para aonde o CPU vai saltar. A rotina de Comando CLs, limpa o écran mantendo as cores da MARGEM, do PAPEL e da TINTA, de acordo com os valores encontrados nas respectivas variáveis do Sistema.

De volta à rotina controladora, esta vai passar a interpretar o Grupo Basic seguinte (20 GOTO 50).

Neste caso, a instrução GOTO é um comando cuja localização é encontrada no endereço 6781 da Tábua dos Comandos. Esse endereço indica-nos que o Comando GOTO pertence à classe 6 e que a localização da rotina respectiva é o endereço 7783.

A rotina do Comando GOTO identifica o valor do operando (no nosso exemplo, o decimal 50), que vai introduzir na variável do Sistema NEW PPC.

Esta variável **informa** a rotina controladora, qual o próximo número de linha a ser interpretado.

Se o operando deste comando, em vez de uma expressão numérica simples (o número 50) fosse uma expressão complexa (exemplo: 100 *b + 1000), é óbvio que teria primeiro de ser calculada.

ROTINA AVALIADORA DE EXPRESSÕES — Está localizada nos endereços 9467 a 10417.

No Spectrum, o resultado obtido na avaliação duma expressão pode ser um resultado **numérico** ou um resultado **String.**

Um resultado numérico é devolvido pela rotina avaliadora, em numeração de ponto flutuante (a 5 bytes), e colocado no topo do Stack do Calculador.

Numa expressão String, o resultado devolvido é igualmente de 5 bytes, que representam os parâmetros que descrevem a String.

As expressões são avaliadas da esquerda para a direita e as diferentes operações matemáticas (se as houver), executadas atendendo às regras de prioridade descritas no capítulo 7 do Manual Spectrum.

Certas operações, tais como, FN, RND, PI, INKEY$, BIN, SCREEN$, ATTR e POINT são normalmente tratadas dentro da rotina avaliadora. Muitas outras, terão de ser tratadas nas rotinas do Calculador.

Quando uma variável do Basic é usada numa expressão, a rotina avaliadora vai recolher o valor dessa variável, chamando a Rotina de Identificação de variáveis.

ROTINAS DE TRATAMENTO DAS VARIÁVEIS — Estas rotinas ocupam as localizações 10418 a 10955. A sua missão consiste na pesquisa de uma variável, para encontrar o seu valor actualizado, analisando para tal, toda a área de variáveis relativas ao programa.

No caso de uma variável array, todos os elementos referentes à sua posição têm de ser identificados antes de autorizada a transferência do seu conteúdo.

Quando se trata duma variável String fraccionada, os parâmetros são devidamente analisados antes de ser transferida a fracção requerida.

AS ROTINAS ARITMÉTICAS — Estas rotinas ocupam os endereços 11400 a 12186. As mais destacáveis deste grupo serão talvez a rotina STACK-BC localizada em 11563, que converte o valor contido no registro BC em numeração de ponto flutuante e o coloca no topo do Stack do calculador, e a rotina PRINT-FP localizada em 11747, que copia o valor armazenado no topo do Stack do calculador, convertendo-o num número decimal para o escrever no écran ou na Printer.

ROTINA DO CALCULADOR — Esta grande e complicada rotina ocupa os endereços 12187 a 14445. Normalmente é chamada através da instrução RST 40 que promove um salto indirecto para a localização 13147.

O calculador possui 66 subrotinas, cada uma destinada a executar uma diferente operação matemática. Para chamar qualquer uma subrotina, o calculador não utiliza a habitual instrução CALL (N+256 * N), mas apenas 1 byte de informação denominado **literal** e uma tábua de endereços localizada em 13015.

Não é invulgar o calculador necessitar de chamar 15 ou 20 subrotinas, entre operações e funções, até ser encontrada a resolução duma expressão complexa.
Usar apenas um byte DATA para chamar cada uma dessas subrotinas é sem dúvida um processo muito engenhoso. Note o leitor que, para além de poupar muitos bytes de ROM, o tempo de cálculo é substancialmente reduzido.
Os **literais** são assemblados como DATA em DEFB(s) — bytes definidos, colocados em sequência operacional logo a seguir à instrução RST 40. Como último byte definido, o literal 56 assegura a execução da operação **Fim de Cálculo** e o retorno do calculador.

EXEMPLO 1 — **Multiplicação de 2 números**

Dois valores são colocados previamente no topo do Stack do calculador, utilizando por exemplo a rotina STACK-BC. Vão ser multiplicados na rotina respectiva do calculador e introduzido o resultado no écran por acção da rotina PRINT-FP.

```
ORG 65400     } Start da rotina
LD BC 450     } 1.º valor é colocado
CALL 11563    } no topo do Stack calculador
LD BC 20      } Idem para o 2.º valor
CALL 11563    }
RST 40          Salto para o calculador
DEFB 4          Subrotina de multiplicação
DEFB 56       } Fim de cálculo e retorno
LD A 2        } Abrir o canal «S»
CALL 5633       Autorização de escrever no écran
CALL 11747      Transferência do Stack para o écran
RET
```

EXEMPLO 2 — **Função TANGENTE de (x)**

O valor (x) é colocado previamente no Stack do calculador. Por acção da subrotina TAN FUNCTION, endereçada em 14298 (cuja cópia aparece neste exemplo), o valor (x) é substituído no Stack pelo resultado da função.

```
ORG 65400       Start do programa
LD BC x         Valor de (x) em radianos
CALL 11563      Coloca x no Stack
RST 40          Calculador. (x) é lido do Stack
DEFB 49         Duplica (x). (x, x)
DEFB 31         SIN x (x, SIN x)
DEFB 1          Troca valores (SIN x, x)
DEFB 32         COS x (SIN x, COS x)
DEFB 5          Divisão (SIN x/COS x)
DEFB 56         Fim de cálculo — resultado no Stack
LD A 2        } Abrir o canal «S»
CALL 5633     }
CALL 11747      Transferência
RET
```

Ensaios mais amplos sobre o calculador e a suas subrotinas são apresentados na devida altura.

BLOCO 3 — A TABELA DE CARACTERES

Está localizada nos endereços 15616 a 16383. Uma área não utilizada da ROM, endereços 14446 a 15615, isola a TÁBUA DE CARACTERES da restante parte do monitor.
A tábua contém os DATA para a formação dos 96 caracteres que o Spectrum utiliza, em **placas** de 8 * bits.
O exemplo a seguir mostra-nos o formato do caracter «+».

Endereços 15704 a 15711

```
15704   DEFB 0    00000000
15705   DEFB 0    00000000
15706   DEFB 8    00001000
15707   DEFB 8    00001000
15708   DEFB 62   00111110
15709   DEFB 8    00001001
15710   DEFB 8    00001000
15711   DEFB 0    00000000
```

A seguir um programa Basic, que lhes mostra a constituição dos 96 caracteres.

```
 10 REM IMAGEM DA TÁBUA DE CARACTERES
 20 FOR A = 15616 TO 16376 STEP 8
 30 FOR B = 0 TO 7 : LET F = PEEK (A + B):
    GOSUB 300 : NEXT B
 40 PRINT
 50 PRINT TAB 2 ; «'» ; CHR$ (32+(A-15616)/8); «'»
 60 INPUT «PRIMA UMA TECLA» ; A$
 70 CLS
 80 NEXT A
 90 STOP
300 REM BINÁRIO DE F
310 FOR N=7 TO 1  STEP — 1
320 LET P=2 ↑ N
330 PRINT CHR$ (48+INT (F/P));
340 LET F=-INT (F/P) * P
350 NEXT N
360 PRINT INT F
370 RETURN
```

(Continua)

UMA VIAGEM PELO INTERIOR DO SPECTRUM

(Continuação do número anterior)

Terminados alguns esclarecimentos sobre as funções do CPU, vamos incluir no texto o diagrama bloco do Spectrum que dará ao leitor uma noção de princípio, das interligações entre os vários componentes cujos esquemas e detalhes de funcionamento falaremos a seguir.

3 — Memórias RAM (comuns aos Spectruns de 16 e 48 K).

Este grupo de memórias é interligado, nas 2 versões do Spectrum, conforme esquema da figura 3.1. O Spectrum de 48 k tem um grupo adicional de memórias RAM de que falaremos mais adiante.

Neste grupo, podem ser armazenados todos os dados necessários à formação da imagem no écran da TV, os vários tipos de variáveis indispensáveis ao Basic, o espaço de trabalho, o espaço de programa, etc.

Cada **um** dos 8 integrados que formam o conjunto, contém 128 FILAS a 128 COLUNAS de células bit.

128 filas * 128 colunas = 16384 células bit

16384/8 = 2048 bytes ou 2 k

Portanto, os 8 integrados (IC7 a IC14) têm uma capacidade de armazenamento de 16 k bytes (16384 bytes).

Cada integrado é responsável para elaboração duma linha do BUS de Dados, pelo que os endereços encaminhados pelo BUS de endereços, têm de sofrer um tratamento elaborado (integrados IC3 e IC4).

Como já fizemos referência, o BUS de Endereços do Spectrum é formado por 16 linhas (A0 a A15), que podem encaminhar (2^{16} = 65536) endereços diferentes.

Pelo esquema da figura 3.1, podemos observar que este grupo de memórias está ligado a 14 linhas do BUS de endereços (A0 a A13).

Assim, 2^{14} =16384) endereços diferentes podem ser encaminhados para as memórias, mas o processo é um pouco complicado. Vamos analisá-lo?

3 sinais enviados pela ULA comandam as operações do Sistema.

a) O sinal RAS, que tem duas funções. A primeira é um sinal de permissão (ligado aos pinos 1 dos integrados IC3 e 4 através da resistência R32 — 10052) que estando presente

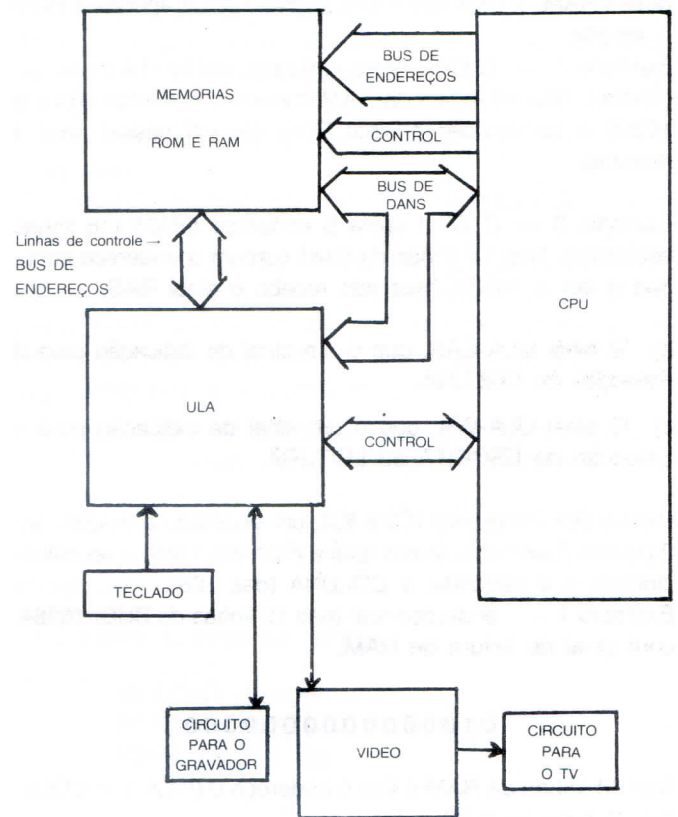



FIG 2.2 — DIAGRAMA BLOCO DO SPECTRUM

autoriza a passagem do endereço para os integrados de memória (IC7 a IC14). A segunda, um sinal de indicação para a selecção de FILA desses integrados.

O sinal de permissão é indispensável para informar a RAM que o endereço presente no BUS lhe é destinado. Note o leitor que endereçada antes da RAM está a ROM cujos endereços são de 0 a 16383.

Como podemos observar no diagrama acima, a RAM comum cobre 16384 endereços, deslocados de zero pelos 16384 endereços da ROM.

Sempre que o endereço colocado no BUS tem o valor de (16384 a 32767) o sinal de permissão RAS autoriza a sua passagem para os integrados da memória.

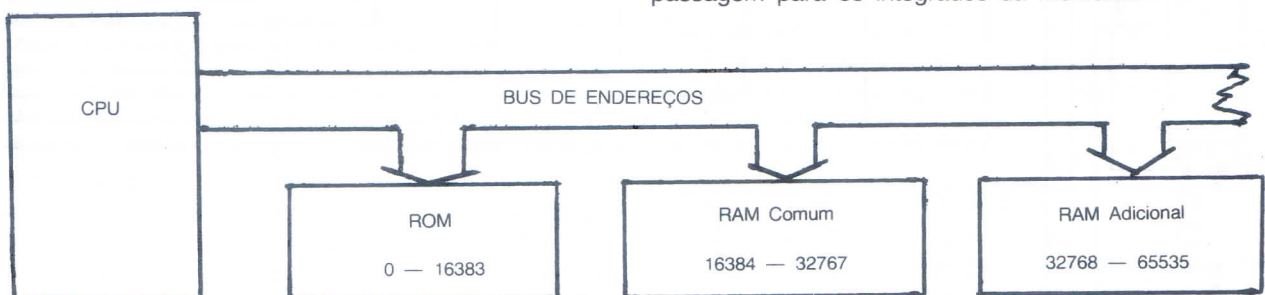



FIG. 3 — diagrama do endereçamento das memórias

Exemplo 1 — O CPU envia um endereço que excita (coloca no valor 1) 14 das 16 linhas do BUS. O valor decimal desse endereço (2^{14}—1 = 16383) que é um endereço para a ROM, vai também estar presente nas 14 linhas do BUS interligadas com a RAM comum. Como o endereço está fora da gama dessa RAM, a ULA não excita o sinal RAS e apenas a ROM o recebe.
Exemplo 2 — O CPU envia endereço 32767 (15 linhas excitadas). Nas 14 linhas da RAM comum o endereço lido é o 16383 e ao receber o sinal RAS, ele vai passar para a memória.

Exemplo 3 — O CPU envia o endereço 65535 (16 linhas excitadas). Nas 14 linhas da RAM comum o endereço continua a ser o 16383, mas não recebe o sinal RAS.

b) O sinal ULA-CAS, que é um sinal de indicação para a Selecção de COLUNA.

c) O sinal ULA-WR, que é um sinal de indicação para a Seleccão de ESCRITA ou LEITURA.

Dentro dos integrados IC3 e IC4, um endereço é dividido em 2 partes. A primeira, indica qual a FILA das (128) a ser seleccionada e a segunda, a COLUNA (das 128).
Exemplo 1 — endereço real (nas 16 linhas do BUS) 16384, com sinal de leitura de RAM.

0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0

Nas 14 linhas da RAM é lido o endereço 0 (FILA 1 — COLUNA 1) para os 8 integrados.

Do integrado IC 9 sai o bit 0 do byte DATA
Do integrado IC 7 sai o bit 1 do byte DATA
Do integrado IC 8 sai o bit 2 do byte DATA
Do integrado IC 10 sai o bit 3 do byte DATA
Do integrado IC 11 sai o bit 4 do byte DATA
Do integrado IC 12 sai o bit 5 do byte DATA
Do integrado IC 13 sai o bit 6 do byte DATA
Do integrado IC 14 sai o bit 7 do byte DATA

No BUS de Dados (D0 a D7) ficará disponível o Byte armazenado no endereço 0 da RAM, correspondente ao endereço real 16384.
Para a elaboração dos sinais de vídeo, a ULA pelas linhas DRAM 0 a DRAM 6 e utilizando o mesmo princípio de selecção já esclarecido, pode obter dados da RAM enviando os endereços adequados. No Spectrum o Ficheiro de Projecção fica armazenado na RAM nos endereços 16384 a 23295.
As resistências R17 a R23 de 330 Ω, isolam as tensões de saída dos integrados IC3 e IC4 das tensões enviadas pela ULA, quando esta selecciona endereços através das linhas DRAM.
Do **entendimento** entre o CPU e a ULA para determinar a cada momento quem vai enviar um endereço, falaremos mais adiante.

(Continua no próximo número)

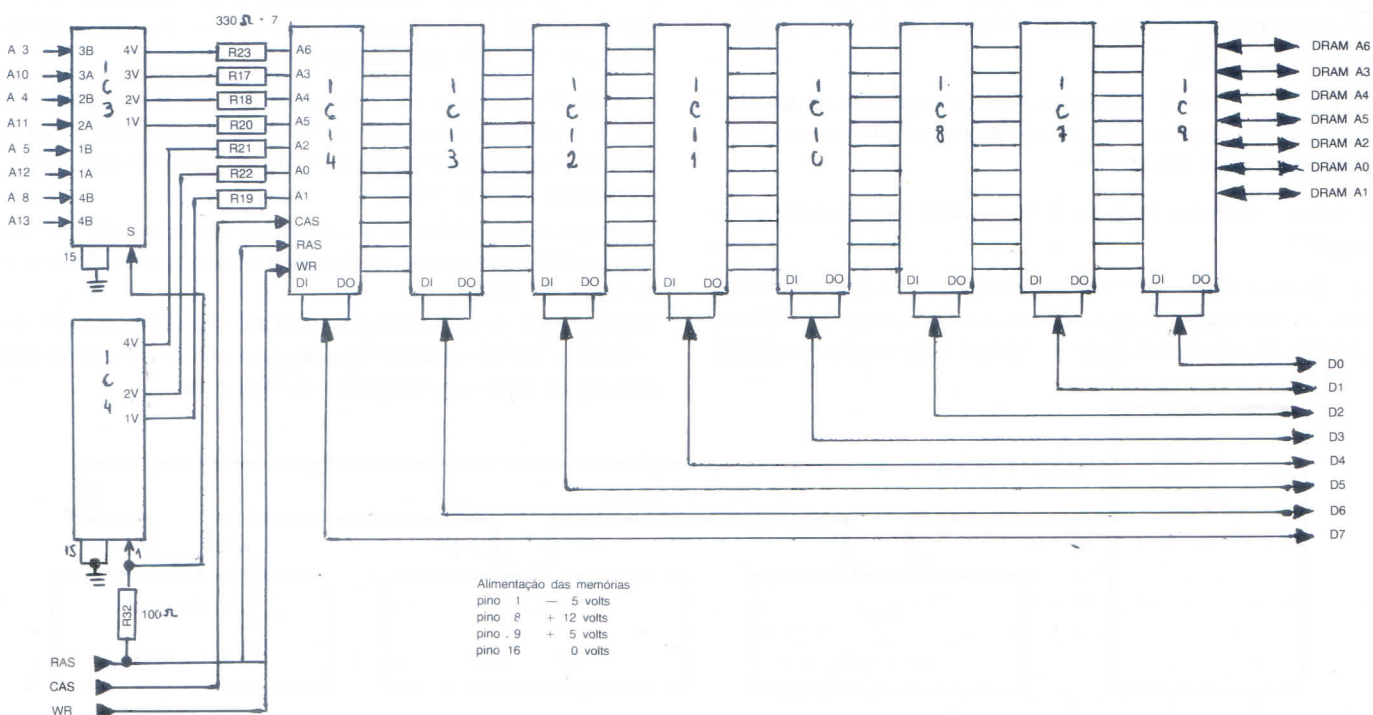



FIG. 3.1 — Esquema da RAM comum

BLAST

BLAST é o primeiro COMPILADOR BASIC inteiramente compatível, em qualquer computador Sinclair. O seu princípio é de dar o máximo de rendimento possível a programas escritos no Basic do Spectrum. Blast pode incrementar a velocidade do Basic elevando-a a um factorial 40.
O compilador e extremamente fácil de utilizar; existem novos comandos que é necessário aprender, para se trabalhar com ele. O nível de compatibilidade com o interpretador e tão alto que até as extensões de código máquina para o Basic, serão compiladas com sucesso.
Juntamente com o Blast vem o programa TOOLKIT, que lhe irá permitir executar uma série de procedimentos (este programa não pode residir na RAM ao mesmo tempo que o compilador BLAST).
Blast foi desenvolvido pela mesma companhia que produziu PETSPEED, o compilador Basic para o Commodore 64, e outro tipo de compiladores para outras máquinas; produziu também OXFORD PASCAL, uma implementação completa da popular linguagem PASCAL, para muitos micros incluindo o Spectrum.
Esperamos que consiga disfrutar plenamente, de todas as possibilidades oferecidas por este poderoso compilador. Juntamente com o programa (cassette) irá um manual de instruções para o seu funcionamento.
Embora a maioria dos utilizadores do Blast possuam bons conhecimentos de Basic, qualquer principiante poderá trabalhar com o BLAST, sem que tenha grandes conhecimentos de Basic. Antes de utilizar o Blast, faça um estudo atento ao manual.

Como utilizar o BLAST

Utilizamos um programa em Basic, já instalado no computador compilámo-lo directamente na RAM sem qualquer gravador ou microdrive.
O programa de recurso e o programa objecto devem estar na memória ao mesmo tempo.
Para fazer o LOAD do programa BLAST, o seguinte procedimento:

LOAD «BLAST» (enter)

Quando o programa começar a ser lido, aparecerá no écran a seguinte mensagem:

BLAST (c) OCSS 1985 XXXXX BYTES FREE

esta mensagem só lhe aparece, quando faz a primeira vez o LOAD do BLAST; uma vez instalado, Blast permite-lhe desejar quantos programas desejar.
Dentro do manual ou numa folha separada encontrará uma matriz de quadrados, cada quadrado pode ser identificado por uma simples grelha de referência. Por exemplo para encontrar o quadrado E-13, basta identificar a coluna E, fila 13. O quadrado E-13 será onde a coluna e a respectiva fila se encontram. O código (senha) da protecção é muito simples; tudo o que deverá fazer é identificar correctamente 4 quadrados e introduzir as suas cores.
BLAST dar-lhe-á instruções como a seguinte:

ENTER THE COLOUR IN SQUARE X.XX (W.Y.G.R.)?

Onde X.XX é a grelha de referência, quando tiver encontrado o quadrado introduza uma das letras acima indicadas (W.Y.G.R.) dependendo da cor do quadrado (W-branco, Y-amarelo, G-verde, R-vermelho) e seguidamente faça ENTER.
Blast encontra-se agora instalado e pronto para compilar programas.
Os comandos do Blast encontram-se procedidos de um asterisco *.
Se escrever *C; Blast irá compilar o programa que se encontra em memória. Para fazer o RUN da versão compilada, escreva *R (ser-lhe-á informado a retirada da memória do programa que não está compilado, se não quiser que isso aconteça accione N, caso contrário Y).

Se quiser fazer a cópia de programas compilados pelo BLAST, não poderá utilizar o comando SAVE, faça da seguinte maneira:

1) Faça o LOAD do programa compilado, pelo Blast pelo computador.

2) Escreva as seguintes linhas:

```
15 LOAD «(prog)»
20 RANDOMIZE USR PEEK 23635 + 256
PEEK 23636 + 150
```

3) escrever

```
SAVE «(prog)» LINE 15
```

Poderá então obter uma cópia do seu programa compilado, para verificar se está correcta; deverá substituir SAVE por VERIFY e proceder da mesma maneira.

GRAVAR PARA MICRODRIVE

O procedimento para gravar em microdrive e exactamente da mesma forma, terá no entanto que obedecer a certos parâmetros específicos de microdrive:

```
15 LOAD * «m;1;« (prog)
20 RANDOMIZE USR PEEK 23635 + 256 *
* PEEK 23636 + 150
```

Depois escreva SAVE * «m;1;« (prog)» LINE 15

Tenha sempre presente que se quiser eliminar uma versão em Basic da memória, nunca utilize NEW, porque o Blast também será eliminado, deverá utilizar como já foi dito *N.

ANÚNCIO

Para satisfazer o pedido de um amigo, dirigimo-nos aos nossos associados no sentido de obter as instruções em português ou inglês do programa SOFKIT TOOLKIT — GRAFKIT.

CICLOS ILUMINADOS NO SEU SPECTRUM

POPULAR COMPUTING

— ciclos iluminados no seu Spectrum

O jogo é 100 % em código máquina, ocupando aproximadamente 1/2 k de memória, incluindo o screen.

Quando o jogo começa (início da primeira das cinco victórias), o programa regressa ao Basic, e pede-lhe um novo nível de perícia.

Primeiro escreva o programa 1 e grave-o em gravador com SAVE "1" LINE 20.

Depois escreva o programa 2 (código máquina) e faça RUN. O código gerado deve ser gravado a seguir ao programa 1, com SAVE "2" CODE 40000,582.

Finalmente escreva o programa 3 e grave-o directamente a seguir ao código máquina com SAVE "3" LINE 15.

Agora faça NEW (elimnando os programas), ponha de novo a cassete no lugar onde começou a gravar o primeiro programa, e faça LOAD da maneira que normalmente costuma fazer para a leitura de qualquer programa. Depois de reproduzido o programa fará o arranque automático.

COMANDOS

Jogador 1	Jogador 2
Z — Esquerda	N — Esquerda
X — Direita	M — Direita
C — P/Baixo	SPACE — P/Baixo
F — P/Cima	ENTER — P/Cima

```
  10 REM ** Programa 1 **
  20 CLEAR 39999
  30 LOAD ""CODE
  40 LOAD ""
```

```
  10 REM ** Programa 2 M/C LOADE
R **
  15 CLEAR 39999
  20 FOR i=40000 TO 40581: READ
a: POKE i,a: NEXT i
  40 STOP
 100 DATA 175,50,156,99,50,157,9
9,50,8,92,62,1,50,162,99,62
 110 DATA 2,50,163,99,62,10,50,1
58,99,50,160,99,205,107,13
 120 DATA 62,2,205,1,22,62,11,50
,159,99,62,21,50,161,99,62,16,21
5,62,2,215,62,17,215
 130 DATA 62,7,215,6,30,62,22,21
5,175,215,120,215,62,144,215,62,
22,215,62,20,215
 140 DATA 120,215,62,144,215,16,
233,6,20,62,22,215,62,1,215,62,1
45,215,62,22
 150 DATA 215,120,215,62,30,215,
62,145,215,16,282,62,17,21,62,1,
215,205,25,158,205
 160 DATA 97,158,205,223,157,62,
16,215,62,6,215,62,22,215,58,158
,99,215,58,159,99,215
 170 DATA 62,146,215,62,16,215,6
2,4,215,62,22,215,58,160,99,215,
58,161,99,215,62,147
 180 DATA 215,33,162,29,58,8,92,
254,122,202,105,157,254,110,202,
125,157,254,120,202
 190 DATA 110,157,254,109,202,13
1,157,254,99,202,120,157,254,32,
202,143,157,254,102,202
 200 DATA 115,157,254,13,202,137
,157,58,162,99,254,1,202,149,157
,254,2,202,159,157,254
 210 DATA 3,202,169,157,58,159,9
9,60,50,159,99,58,163,99,254,1,2
02,179,157,254,2,202
 220 DATA 189,157,254,3,202,199,
157,58,161,99,60,50,161,99,58,15
8,99,71,58,159,99
 230 DATA 79,205,209,159,126,254
,9,194,243,57,58,160,99,61,58,16
1,99,79,205,209,157
 240 DATA 126,254,9,194,1,158,19
5,165,156,54,3,195,18,157,54,4,1
95,18,157,54,1,195,18
 250 DATA 157,54,2,195,18,157,35
,54,3,195,18,157,3,54,4,195,18,1
57,35,54,1,195,18,157
 260 DATA 35,54,2,195,18,157,58,
158,99,61,50,158,99,195,43,157,5
8,158,99,60,50,158
 270 DATA 99,195,43,157,58,159,9
9,61,50,159,99,195,43,157,58,160
,99,61,50,160,99,195
 280 DATA 68,157,58,160,99,60,50
,160,99,195,68,157,58,161,99,61,
50,161,99,195,68,157
 290 DATA 33,0,88,17,32,0,25,16,
253,22,0,89,25,201,58,164,99,198
,4,71,118,16,253
 300 DATA 201,205,223,157,205,22
3,157,205,223,157,58,157,99,60,5
0,157,99,254,5,202,25
 310 DATA 158,195,71,156,205,223
,157,205,223,157,58,156,99,60,50
,156,99,154
 320 DATA 5,202,25,158,195,71,15
6,62,16,215,62,5,215,62,22,215,6
2,21,215,62,14,215
 330 DATA 6,0,58,156,99,79,205,2
7,26,62,21,215,62,1,215,6,13,62,
32,215,16,215,62
 340 DATA 21,215,175,215,58,157,
99,79,205,27,26,201,80,76,65,89,
69,82,46,49,32,32,32
 350 DATA 32,32,32,80,76,65,89,6
9,82,46,50,62,21,215,62,1,215,62
,16,215,62,5,215
 360 DATA 62,22,215,62,21,215,62
,4,215,6,22,33,75,158,126,215,35
,16,251,62,21,215,175,215,201
```

```
   5 REM ** Programa 3 **
  15 FOR i=USR "a" TO USR "d"+7:
 READ a: POKE i,a: NEXT i
  20 FOR i=1 TO 2: BORDER 1: PAP
ER 1: CLS : NEXT i: INK 7
  30 INPUT "NIVEL (1 ate 10) <<1
=MAX>> ";a
  40 IF a<1 OR a>10 THEN GO TO 2
0
  50 POKE 25508,2: INK 1
  60 RANDOMIZE USR 40000
  65 POKE 23561,35: POKE 23562,5
  70 INK 7: PRINT AT 10,10; " FI
NAL DO JOGO"
  80 FOR i=1 TO 1000: NEXT i
  90 GO TO 20
 100 DATA 239,239,239,0,254,254,
254,0
 110 DATA 119,119,119,112,119,11
9,119,7
 120 DATA 60,110,223,191,191,223
,110,60
 130 DATA 60,102,219,189,255,255
,126,60
```

ANÚNCIO

Aproveitamos a deixa, para dizer aos nossos sócios e amigos que todas as sugestões que queiram são sempre bem vindas.

TODOS OS POKES E MAIS ALGUNS

Este programa contém todas as opções para gravar o SCREEN. Se não quiser todos os POKES é uma ideia escrevê-los todos e gravá-los, apagando os que não interessar, quando está a fazer RUN no programa.
As linhas 20 e 30 deixam-no alterar o número de objectos e o compartimento de partida, na linha 40 é definido o número máximo de monstros por compartimento.

```
  10 CLEAR 64999
  20 LET obj=50
  30 LET room=32
  40 LET mons=15
  50 FOR n=65460 TO 65529: READ
a: POKE n,a: NEXT n
  52 DATA 205,84,31,210,152,116,
62,251,219,254,203,103,194,145,1
17,33,3,0,229,205,142,2,205,30,3
,56,248,205,142,2,205,30,3,48,24
,8,246,32,254,115,204
  54 DATA 122,15,214,48,254,10,4
8,27,225,6,10,132,16,253,103,45,
32,216,61,254,134,210,145,117,50
,130,80,195,86,117
  60 FOR n=65000 TO 65044: READ
a: POKE n,a: NEXT n
  62 DATA 221,33,0,64,17,56,185,
62,255,55,205,86,5,243,48,240
  64 DATA 33,176,244,17,176,247,
1,179,143,237,184,62,247,50,164,
100
  66 DATA 33,76,254,17,197,100,6
,1,237,176,195,0,95
  70 LET n=65100
  72 READ a: IF a=999 THEN GO TO
 1000
  76 POKE n,a: LET n=n+1: GO TO
72
  80 DATA 33,221,100,17,121,115,
1,51,0,237,176,33,180,255,34,140
,117,62,195,50,139,117,24,51
  82 DATA 0,221,229,221,33,155,1
15,17,17,0,175,205,198,4,27,122,
79,32,251,221,33,0,64
  84 DATA 22,27,61,205,198,4,221
,225,62,255,201,3,33,10,10,18,1,
74,83,87,32,170,0,27,64,0,128
  90 DATA 62,201,50,248,121: REM
 invencivel
 100 DATA 62,255,50,67,117: REM
vidas infinitas
 110 DATA 62,obj,50,126,135: REM
 n.de objectos para recolher
 120 DATA 62,room,50,75,117: REM
 compartimento de arranque
 130 DATA 62,255,50,115,119: REM
 altura para cair
 140 DATA 62,mons,50,169,123: RE
M n.maximo de monstros
 150 DATA 62,0,50,222,131: REM c
omecar a andar
 160 DATA 33,254,3,34,227,118,33
,220,13,34,229,118,33,123,118,33
,123,62,34,231,118,33,4,40,34,23
3,118,62,8,50,235,118: REM levan
tar objectos
 999 DATA 195,0,112,999
1000 PAPER 0: INK 0: BORDER 0: C
LS
1010 RANDOMIZE USR 65000
```

DAY SPRITE DESIGNER

«Your Spectrum» — Setembro, n.º 18

Escreva esta pequena listagem e faça RUN terá então uma Sprite especial de desenho, para a utilizar precisa das seguintes instruções:

Q — Move o cursor para cima
A — Move o cursor para baixo

O — Move o cursor para a esquerda
P — Move o cursor para a direita

Se accionar a tecla SPACE, o PIXEL se estiver branco pasará a preto e vice-versa. Para parar accione a tecla F.

```
  10 CLS : PRINT "SPRITE DESIGNE
R"
  20 FOR f=0 TO 24*6 STEP 6: PLO
T f,0: DRAW 0,16*6: NEXT f
  30 FOR f=0 TO 16*6 STEP f: PLO
T 0,f: DRAW 24*6,0: NEXT f
  40 PLOT 182,80: DRAW 0,19: DRA
W 27,0: DRAW 0,-19: DRAW -27,0
  50 LET x=0: LET y=0
 100 LET x=x+(INKEY$="p" AND x<2
3)-(INKEY$="o" AND x>0)
 110 LET y=y+(INKEY$="q" AND y<1
5)-(INKEY$="a" AND y>0)
 120 GO SUB 9000: GO SUB 9000
 130 IF INKEY$=CHR$ 32 THEN GO S
UB 9000
 135 IF INKEY$=CHR$ 32 THEN GO T
O 135
 140 IF INKEY$="f" THEN STOP
 150 GO TO 100
9000 OVER 1: PLOT x+184,y+82: FO
R f=y*6+1 TO y*6+5: PLOT x*6+1,f
: DRAW 4,0: NEXT f: OVER 0: RETU
RN
```

RUN FOR IT

«YOUR SPECTRUM» — Setembro n.º 18

Apresentamos um pequeno programa de RUN para microdrives que oferece uma sintaxe simplificada, para reproduzir as files.
Em vez de escrever LOAD*"m";1; "nome do programa" accione a tecla NEW e depois faça RUN : REM nome do programa.
O programa faz o arranque automático. Se houver qualquer problema será reproduzida uma mensagem de erro do tipo «nonsense in Basic». Antes de gravar faça CLEAR, para ter a certeza que a área ocupada pelas variáveis está vazia.

```
  10 PRINT "MLJNW 1985": LET a$=
"          ": LET b=0: LET c=0
  20 LET v=PEEK 23672+256*PEEK 2
3628
  30 LET a=PEEK 23641+256*PEEK 2
3642
  40 FOR n=3 TO 12
  50 LET b=PEEK (a+n+25): IF b=1
3 THEN GO TO 70
  60 POKE v+n,b: NEXT n
  70 PRINT "LEITURA :";a$
  80 LOAD *"m";1;a$
```

ENCONTRE OS COMPARTIMENTOS

Está perdido, confuso no labirinto?
Deixe que estes dois programas o ajudem, o primeiro grava o número de quartos e as suas saídas, escreva-o, faça RUN não se esqueça de accionar o gravador, para gravar a DATA antes de fazer ENTER.
O segundo programa faz o LOAD e imprime a DATA na sua impressora. Na linha 50 pode mudar o LPRINT em PRINT, se não quiser a impressora.

```
   5 REM ** PRIMEIRO **
  10 INK 0: PAPER 0: BORDER 0: C
LEAR 65439: FOR n=65440 TO 65511
: READ a: POKE n,a: NEXT n
  20 RANDOMIZE USR 65440
  30 DATA 221,33,0,64,17,56,185,
62,255,55,205,86,5,243,48,240,33
,185,255,34,198,100,195,5,95
  40 DATA 5,134,17,0,64,33,252,1
86,197,35,126,35,229,102,198,12,
111,48,1,36,205,77,128,43,1,5,0,
237,176
  50 DATA 225,193,16,231,221,33,
0,64,17,246,9,175,205,198,4,195,
0,112
```

```
   5 REM ** SEGUNDO **
  10 CLEAR 39999: FOR n=50000 TO
 50011: READ a: POKE n,a: NEXT n
: RANDOMIZE USR 50000: LET l=400
00
  20 FOR n=1 TO 134: LET a$=""
  30 LET a=PEEK l: LET l=l+1: IF
 a>127 THEN LET a=a-128: IF PEEK
 l>127 THEN LET a$=a$+CHR$ a: GO
 TO 50
  40 LET a$=a$+CHR$ a: GO TO 30
  50 DIM b$(32): LET b$=a$: LPRI
NT "b$;n;":";"Lt:";PEEK (l+1);"U
P:";PEEK (l+2);"Rt:";PEEK (l+3);
"Dn:";PEEK (l+4): LET l=l+5: NEX
T n
  60 STOP
 100 DATA 221,33,64,156,17,246,9
,175,55,195,2,8
```

MEGADRIVIN

«Your Spectrum — Setembro, n.º 18

Para quem possui microdrives e o utilitário Megabasic, apresentamos aqui uma maneira fácil de definir as chaves para gravar, ler, apagar, etc.

KEY — 1, "INPUT""SAVE FILENAME:"";a$: SAVE*""m"; 1;a$"chr$13

Para usar isto, faça-o em "EXTENDED MODE" e accione Symbol Shfit/1.
Deverá seguidamente indicar o nome da file e o programa será automáticamente gravado em microdrive.

Deve fazer o mesmo para a reproduzir, formatar, etc., definindo a chave com Load ou Format, ou outro comando conforme o caso.

Para gravar a sua versão do Megabasic, introduza o programa Megabufix (publicado na revista de Julho), faça o Load e o Spectrum fará seguidamente NEW, grave com SAVE * "m";1; "MB" CODE 44996,20373 (não se esqueça de apagar primeiro o código antigo) e cada vez que fizer o Load, as chaves estarão lá definidas.

POKES EM ABUNDÂNCIA

«Your Spectrum — Setembro, n.º 18

Esta primeira listagem é para aqueles que querem vidas infinitas (nos seus jogos) e nunca conseguem, além disso poderá também escolher o compartimento de arranque e o número de objectos necessários para acabar. Altere os números nas linhas 20 e 30 para os valores que quiser e faça RUN.

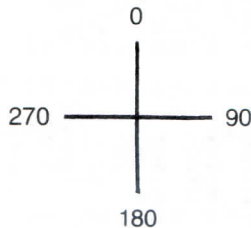
```
  10 CLEAR 64999
  20 LET obj=150
  30 LET room=32
  40 FOR n=65000 TO 65047: READ
a: POKE n,a: NEXT n
  50 PAPER 0: INK 0: BORDER 0: C
LS
  60 RANDOMIZE USR 65000
  70 DATA 221,33,0,64,17,56,185,
62,255,55,205,86,5,243,48,240
  80 DATA 33,6,254,17,197,100,1,
59,0,237,176,195,0,95
  90 DATA 62,255,50,67,117
 110 DATA 62,room,50,75,117
 120 DATA 195,0,112
```

MINI LOGO

Science & Vie Micro n.º 19

Esta linguagem de programação, centrada na criação de gráficos é muito útil. A versão seguinte é composta por instruções de desenho, sendo portanto impossível fixar um texto ou criar procedimentos. No início do programa, o editor afixa uma página que comporta um quadro, que vai ocupar a quase totalidade do écran destinado aos gráficos. Na parte esquerda encontra-se o menu e as instruções do programa. O menu do Mini Logo comporta 9 opções; o cursor encontra-se simbolizado por < > maior que), desloca-se pela tecla 6 para baixo e pela 7 para cima, valida-se qualquer opção fazendo ENTER. Para a programação dispomos de duas opções, uma em modo directo e outra permitindo escrever os procedimentos utilizáveis no modo directo.

As ordens de programação no Mini Logo apresentam-se em duas partes, de um lado instruções e de outro a parte numérica, qualquer instrução no seu conjunto não pode ultrapassar cinco caracteres.
As instruções de deslocamento apresentam-se da seguinte forma:

td — o cursor executa um ângulo para a direita
td — o cursor executa um ângulo para a esquerda

di — põe o cursor na direcção indicada
av — avança o cursor passo a passo

O desenho acima indica a orientação dos ângulos. Estas instruções, assim como as que se seguem, devem ser introduzidas em minúsculas.

ef — limpa o écran
ce — põe o cursor no centro do écran
le — o cursor desloca-se sem deixar o traço no écran
po — acciona o traço

Estas instruções não incluem parte numérica; para executar um programa repetidas vezes, utilizaremos:

rexxx — xxx indica o número de vezes que o programa será executado, quando utilizar esta expressão, deverá colocá-la na primeira linha do programa.
Além destes comandos, Mini - Logo aceita 9 variáveis diferentes que serão assinaladas de v1 a v9.
vnxxx — inicializa a variável vn (em que n pode tomar valores de 1 a 9), no valor xxx (entre o 1999).

Podemos igualmente efectuar as quatro operações clássicas, segundo este exemplo:

v1 45 — inicializa a variável v1 com o valor 45.
+v1 20 — acrescenta-se 20 a v1; v1 tomará por conseguinte o valor 65.
Procede-se da mesma maneira para a subtracção, multiplicação e divisão, para pôr as variáveis a 0 utilizaremos o comando c1.

O menu tem as seguintes opções:

1) PROG — para introduzir o programa em modo directo e executá-lo imediatamente. Poderá aí introduzir as ordens clássicas assim como os procedimentos guardados na opção INST.
2) INST — para escrever os procedimentos que serão assinalados pelo programa como subrotinas. Estas instruções não são executadas directamente. Na passagem pelo modo INST é necessário dar um nome ao procedimento, esse nome não deverá ultrapassar um comprimento de 4 caracteres, para chamar um procedimento no programa (opção PROG do menu), deverá escrever $ nome (nome do procedimento), um procedimento não pode conter o nome do outro.
3) LIST — faz a listagem de um procedimento, é necessário introduzir o nome.
4) SAVE — permite gravar na cassette os procedimentos do utilizador.
5) LOAD — permite a reprodução dos procedimentos gravados em cassette.
6) NOMS — faz a listagem dos procedimentos em memória, à partida o programa conhece 4 procedimentos:

NEW — procedimento de reinicialização
HEXA — permite traçar um hexágono
CARR — permite traçar um quadrado
TRIA — permite traçar um triângulo
7) COPY — faz a cópia do écran na impressora
8) S.EC — grava os gráficos na cassette
9) L.EC — reproduz um gráfico gravado em cassette
Vejamos em seguida alguns exemplos de utilização:

Selecciona-se a opção PROG, valida-se por ENTER, e escrevem-se as seguintes linhas:

re 36 — manda executar o programa 36 vezes
av 40 — faz avançar o cursor 40 passos para cima
td 90 — faz girar o cursor para a direita 90 graus.
av 40
td 90
av 40
td 90
av 40
td 90
td 10 — esta última instrução provoca um deslocamento do cursor
fin — sai da escrita do programa e provoca a sua execução.
Este programa desenha um quadro que se repetirá 36 vezes de maneira deslocada para representar uma rosacea. Durante o desenrolar do programa, veremos que cada instrução em curso será afixada em inverse vídeo, no momento em que a executada.
Exemplo de um programa que faz chamada a um procedimento, note que este programa dá o mesmo resultado que o anterior. Selecciona-se PROG e depois escreve-se:

v1 40
fin
volta-se a seleccionar PROG
re 36
= carr
td 10
fin

Este programa estáá em duas partes porque é necessário fazer a inicialização de uma variável antes de usar carr, e sobretudo antes da ordem de repetição do programa, para reinicializar o editor introduziremos no PROG, o programa seguinte:
= NEW
fin
para obter outro tipo de rosácea ainda no PROG
v1 35
fin
PROG
re 36
= hexa
td 10

Para compreender como utilizar estes procedimentos, aconselhamos a listar um dos procedimentos de base: HEXA, CARR ou TRIA.
Último ponto, em caso de interrupção no program, é necessário, fazê-lo continuar com GOTO 40, para não destruir as diversas variáveis utilizadas.

```
   2 CLS
   5 LET p$=CHR$ 25+"NEW ef ce c
l di180"+CHR$ 20+"carred4 avv1 t
d90 "+CHR$ 20+"triare3 avv1 td12
0"
   7 LET p$=p$+CHR$ 20+"hexare6
avv1 td60"
  10 DIM c$(5)
  15 DIM z$(4)
  20 LET x=150: LET y=87
  25 LET e=1
  27 LET i=0
  28 DIM v(9)
  30 PLOT 41,0: DRAW 0,175: DRAW
 214,0: DRAW 0,-175: DRAW -214,0
  35 LET d=180
  40 LET i$=""
  50 GO SUB 1000
  60 PRINT AT 0,0; INK 3;"Prog:"
  70 FOR n=1 TO 21: INPUT "Instr
ucao NO ";(n);": ";c$
  80 IF c$="fin " THEN GO TO 100
  85 PRINT AT n,0; INK 0;c$(1 TO
 2); INK 1;c$(3 TO 5): LET i$=i$
+c$
  90 NEXT n
 100 LET re=1: LET ff=1
 110 IF i$(1 TO 2)="re" THEN LET
 re=VAL i$(3 TO 5): LET ff=6
 120 FOR r=1 TO re
 130 FOR f=ff TO LEN i$-4 STEP 5
 140 PRINT AT f/5+1,0; PAPER 8;
INK 8; INVERSE 1;i$(f TO f+4)
 150 LET l$=i$(f TO f+4)
 160 LET t$=l$(1 TO 2): LET n$=l
$(3 TO 5)
 161 IF l$(1)="#" AND p$<>"" THE
N GO TO 300
 162 IF n$(1)="v" THEN LET n$=ST
R$ (VAL ("v("+n$(2 TO 3)+")"))
 163 IF t$="en" THEN LET i=VAL n
$
 165 IF t$="ef" THEN CLS : PLOT
41,0: DRAW 0,175: DRAW 214,0: DR
AW 0,-175: DRAW -214,0
 170 IF t$="di" THEN LET d=VAL n
$
 172 IF t$="+v" THEN LET v(VAL n
$(1))=v(VAL n$(1))+VAL n$(2 TO 3
)
 174 IF t$="-v" THEN LET v(VAL n
$(1))=v(VAL n$(1))-VAL n$(2 TO 3
)
 176 IF t$="*v" THEN LET v(VAL n
$(1))=v(VAL n$(1))*VAL n$(2 TO 3
)
 178 IF t$="/v" THEN LET v(VAL n
$(1))=v(VAL n$(1))/VAL n$(2 TO 3
)
 179 IF t$(1)="v" THEN LET v(VAL
 t$(2))=VAL n$
 180 IF t$="ce" THEN LET x=150:
LET y=87: GO TO 235
 185 IF t$="cl" THEN DIM v(9): G
O TO 235
 190 IF t$="td" THEN LET d=d+VAL
 n$: GO TO 235
 200 IF t$="tg" THEN LET d=d-VAL
 n$: GO TO 235
 210 IF t$="le" THEN LET e=0: GO
 TO 235
 220 IF t$="po" THEN LET e=1: GO
 TO 235
 230 IF t$="av" THEN LET a=SIN (
d*(PI/180)): LET b=COS (d*(PI/18
0)): PLOT INK i;x,y: LET x=x+(a*
VAL n$): LET y=y+(b*VAL n$): IF
x<255 AND x>41 AND y<175 AND y>0
 THEN DRAW INVERSE NOT e; INK i;
a*VAL n$,b*VAL n$
 235 PRINT AT f/5+1,0; PAPER 8;
INK 8; INVERSE 0;i$(f TO f+4)
 240 NEXT f: NEXT r: FOR n=0 TO
21: PRINT AT n,0;"     ": NEXT n
 250 GO TO 40
 300 LET lect=1
 310 PRINT AT 0,0; INK 2;"Instr"
 320 IF p$(lect+1 TO lect+4)<>l$
(2 TO 5) THEN LET lect=lect+CODE
 (p$(lect))
 330 IF lect>LEN p$ THEN GO TO 7
10
 340 IF p$(lect+1 TO lect+4)<>l$
(2 TO 5) THEN GO TO 320
 350 PRINT AT 0,0; INK 6; PAPER
2;"#"; FLASH 1;p$(lect+1 TO lect
+4)
 360 LET a$=p$(lect+5 TO lect+CO
DE (p$(lect))-1)
 370 LET re2=1: LET ff2=1
 380 IF a$(1 TO 2)="re" THEN LET
 re=VAL a$(3 TO 5): LET ff2=6
 420 FOR s=1 TO re
 430 FOR t=ff2 TO LEN a$-4 STEP
5
 450 LET b$=a$(t TO t+4)
 460 LET d$=b$(1 TO 2): LET e$=b
$(3 TO 5)
 461 IF e$(1)="v" THEN LET e$=ST
R$ (VAL ("v("+e$(2 TO 3)+")"))
 462 IF d$="en" THEN LET i=VAL e
$
 465 IF d$="ef" THEN CLS : PLOT
41,0: DRAW 0,175: DRAW 214,0: DR
AW 0,-175: DRAW -214,0
 470 IF d$="di" THEN LET d=VAL e
$
 472 IF d$="+v" THEN LET v(VAL e
$(1))=v(VAL e$(1))+VAL e$(2 TO 3
)
 474 IF d$="-v" THEN LET v(VAL e
$(1))=v(VAL e$(1))-VAL e$(2 TO 3
)
 476 IF d$="*v" THEN LET v(VAL e
$(1))=v(VAL e$(1))*VAL e$(2 TO 3
)
 478 IF d$="/v" THEN LET v(VAL e
$(1))=v(VAL e$(1))/VAL e$(2 TO 3
)
 479 IF d$(1)="v" THEN LET v(VAL
 d$(2))=VAL e$
 480 IF d$="ce" THEN LET x=150:
LET y=87: GO TO 700
 485 IF d$="cl" THEN DIM v(9): G
O TO 700
 490 IF d$="td" THEN LET d=d+VAL
 e$: GO TO 700
 500 IF d$="tg" THEN LET d=d-VAL
 e$: GO TO 700
 510 IF d$="le" THEN LET e=0: GO
 TO 700
 520 IF d$="po" THEN LET e=1: GO
 TO 700
 530 IF d$="av" THEN LET a=SIN (
d*(PI/180)): LET b=COS (d*(PI/18
0)): PLOT INK i;x,y: LET x=x+(a*
VAL e$): LET y=y+(b+VAL e$): IF
x<255 AND x>41 AND y<175 AND y>0
 THEN DRAW INVERSE NOT e; INK i;
a*VAL e$,b*VAL e$
 700 NEXT t: NEXT s
 710 PRINT AT 0,0; INK 3;"prog:"
 715 PRINT AT f/5+1,0; PAPER 8;
INK 8; INVERSE 0;i$(f TO f+4)
 720 NEXT f: NEXT r
 730 FOR n=0 TO 21: PRINT AT n,0
;"     ": NEXT n
 740 GO TO 40
1000 PRINT AT 0,0;"Menu:"
1010 PRINT AT 1,1;"Prog";AT 2,1;
"Inst";AT 3,1;"List";AT 4,1;"Loa
d";AT 5,1;"Save";AT 6,1;"Noms";A
T 7,1;"Copy";AT 8,1;"S.ec";AT 9,
1;"L.ec"
1020 LET ha=1
1030 IF INKEY$="7" AND ha>1 THEN
 LET ha=ha-1
1040 IF INKEY$="6" AND ha<9 THEN
 LET ha=ha+1
```

```
1050 IF INKEY$=CHR$ 13 THEN GO T
O 1070
1055 PRINT AT ha,0;">"
1060 IF INKEY$<>"" THEN GO TO 10
60
1065 IF INKEY$="" THEN GO TO 106
5
1067 PRINT AT ha,0;" "
1068 GO TO 1030
1070 FOR n=0 TO 9: PRINT AT n,0;
"   ": NEXT n
1075 IF ha=1 THEN RETURN
1076 IF ha=7 THEN COPY : GO TO 1
000
1077 IF ha=8 THEN SAVE "Ecran lo
go"SCREEN$ : GO TO 1000
1078 IF ha=9 THEN LOAD ""SCREEN$
 : GO TO 1000
1080 IF ha=3 THEN GO TO 1500
1082 IF ha=6 THEN GO TO 3000
1084 IF ha=4 THEN GO TO 2000
1087 IF ha=5 THEN GO TO 2500
1090 PRINT AT 0,0;"Inst:"
1097 LET x$=""
1100 INPUT "Nom:";z$: PRINT AT 0
,0; INK 1; PAPER 5;"#"; FLASH 1;
z$
1105 LET x$=x$+"  "+z$
1110 FOR n=1 TO 21: INPUT "instr
ucao No";(n);":";c$
1120 IF c$="fin  " THEN GO TO 120
0
1130 PRINT AT n,0; INK 0;c$(1 TO
 2); INK 1;c$(3 TO 5)
1140 LET x$=x$+c$
1150 NEXT n
1200 LET x$(1)=CHR$ (n*5)
1210 FOR n=0 TO 21: PRINT AT n,0
;"   ": NEXT n
1215 PRINT p$=p$+x$
1220 GO TO 1000
1500 PRINT AT 0,0;"List:"
1510 INPUT "No. da instrucao:";g
$
1512 LET g$=g$+"    "
1515 LET g$=g$(1 TO 4)
1520 PRINT AT 0,0;"#";g$
1530 LET lect=1
1550 IF p$(lect+1 TO lect+4)<>g$
 THEN LET lect=lect+CODE (p$(lec
t))
1560 IF lect>LEN p$ THEN FOR n=0
 TO 21: PRINT AT n,0;"   ": NEXT
n: GO TO 1000
1565 IF p$(lect+1 TO lect+4)<>g$
 THEN GO TO 1550
1570 LET f$=p$(lect+5 TO lect+CO
DE (p$(lect))-1)
1580 PRINT AT 0,0; PAPER 5; INK
1;"#"; FLASH 1;g$
1590 FOR n=1 TO LEN f$-4 STEP 5
1600 PRINT AT n/5+1,0;f$(n TO n+
4)
1610 NEXT n
1620 INPUT "Enter para o Menu";
LINE c$
1630 FOR n=0 TO 21: PRINT AT n,0
;"   ": NEXT n
1640 GO TO 1000
2000 PRINT AT 0,0;"Load:";
2010 LOAD "" DATA o$()
2015 LET p$=o$
2020 CLS : GO TO 10
2500 PRINT AT 0,0;"Save:"
2510 DIM o$(LEN p$)
2520 LET o$=p$
2530 SAVE "Load" DATA o$()
2540 PRINT AT 0,0;"     "
2550 GO TO 1000
3000 PRINT AT 0,0;"Noms:"
3010 LET num=1
3020 LET lect=1
3030 IF lect>LEN p$ OR num>21 TH
EN GO TO 3100
3040 PRINT AT num,0; INK 1; PAPE
R 5;"#"; INK 0; PAPER 7;p$(lect+
1 TO lect+4)
3050 LET num=num+1: LET lect=lec
t+CODE (p$(lect))
3060 GO TO 3030
3100 INPUT "Enter para o Menu";
LINE c$
3110 FOR n=0 TO 21: PRINT AT n,0
;"   ": NEXT n
3120 GO TO 1000
```

QUATRO EM LINHA

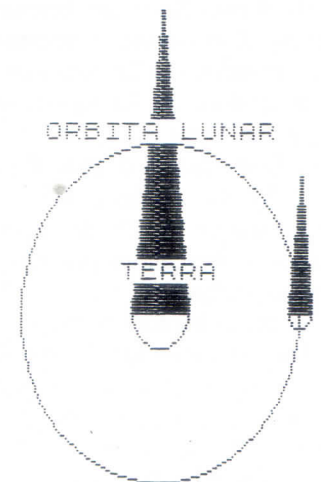
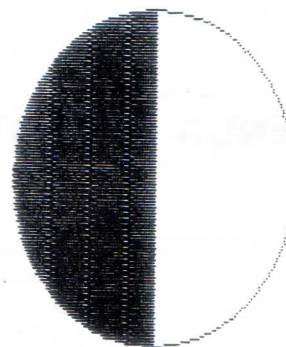
```
   5 REM QUATRO EM LINHA
  10 LET X=1: LET I$="XXX": INK
0: PAPER 7: CLS : DIM A(12,13)
  20 LET S$=CHR$ 144+CHR$ 145+CH
R$ 146
  30 LET T$=CHR$ 147+CHR$ 95+CHR
$ 148
  40 LET U$="                    "
  50 DATA 0,127,127,127,127,127,
127,127
  60 DATA 0,150,150,150,150,150,
150,150
  70 DATA 0,254,254,254,254,254,
254,254
  80 DATA 128,128,128,128,128,12
8,128,255
  90 DATA 1,1,1,1,1,1,1,255
 100 FOR J=144 TO 148: FOR K=0 T
O 7
 110 READ A: POKE USR CHR$ J+K,A
 120 NEXT K: NEXT J
 130 FOR J=2 TO 17 STEP 3
 140 FOR K=2 TO 26 STEP 4
 150 PRINT INK 6; PAPER 0;AT J,K
;S$
 160 PRINT PAPER 6;AT J+1,K;T$
 170 NEXT K: NEXT J
 180 PRINT INK 3;AT 0,3;"1   2
3   4   5   6   7"
 190 PRINT INK 7; PAPER X;AT 20,
10;" JOGADOR ";X;" "
 200 BEEP .2,5: PRINT #0;AT 0,7;
" COLUNA [ 1 A 7 ] ": PAUSE 0
 210 LET A=VAL INKEY$: IF A<>INT
A OR A>7 OR A<=0 THEN GO TO 200
 220 LET C=(A-1)*4+2
 230 IF ATTR (2,C)<>6 THEN BEEP
.2,20: PRINT FLASH 1;AT 21,8;" C
OLUNA ";A;" CHEIA ": PAUSE 100:
PRINT AT 21,8;"               "
: GO TO 200
 240 PRINT AT 21,7;U$
 250 FOR J=0 TO 18
 260 LET Z=ATTR (J,C)
 270 PRINT AT J,C; PAPER X; INK
7;I$
 280 IF J=2 THEN PRINT INK 3;AT
J-2,C;CHR$ 32;A;CHR$ 32
 290 IF J>2 AND Z=6 THEN PRINT P
APER 6;AT J-2,C;T$
 300 IF J>2 AND Z=48 THEN PRINT
AT J-2,C;"   "
 310 IF J>2 AND Z=56 THEN PRINT
INK 6; PAPER 0;AT J-2,C;S$
 320 IF J=18 OR ATTR (J+2,C)=23
OR ATTR (J+2,C)=15 THEN BEEP 0.1
,-20: GO TO 350
 330 IF Z=48 THEN FOR P=1 TO 4:
BEEP 0.05,(20-2*J)+P: NEXT P
 340 NEXT J
 350 LET L=J/3+3: LET C=A+3
 360 LET A(L,C)=X
 370 FOR J=-3 TO 0
 380 IF A(L+J,C)=X AND A(L+J+1,C
)=X AND A(L+J+2,C)=X AND A(L+J+3
,C)=X THEN GO TO 460
 390 IF A(L,C+J)=X AND A(L,C+J+1
)=X AND A(L,C+J+2)=X AND A(L,C+J
+3)=X THEN GO TO 460
 400 IF A(L+J,C+J)=X AND A(L+J+1
,C+J+1)=X AND A(L+J+2,C+J+2)=X A
ND A(L+J+3,C+J+3)=X THEN GO TO 4
60
 410 IF A(L-J,C+J)=X AND A((L-J)
-1,C+J+1)=X AND A((L-J)-2,C+J+2)
=X AND A((L-J)-3,C+J+3)=X THEN G
O TO 460
 420 NEXT J
 430 LET I$="OOO": IF X=2 THEN L
ET I$="XXX"
 440 LET X=X+1: IF X=3 THEN LET
X=1
 450 GO TO 190
 460 PRINT OVER 1; FLASH 1;AT 20
,11;U$( TO 8)
 465 PRINT #0;AT 0,6;"
  "
 470 PRINT AT 20,6; FLASH 1; INK
X;" GANHOU O JOGADOR ";(X);" "
 480 PRINT #0; BRIGHT 1;"    O
UTRO JOGO ?   (S/N)      ": PAUS
E 0
 490 LET A$=INKEY$: IF A$="N" OR
A$="n" THEN STOP
 500 CLS : GO TO 130
 510 PRINT AT 21,0;U$;AT 21,16;U
$
 520 DIM a(12,13): GO TO 130
9990 SAVE "4 EM LINHA" LINE 1
```

FASES DA LUA

```
   5 REM     FASES DA LUA
  ADAPT. MICROHOBBY ANO II - 24
  10 BEEP .2,5: CLS : PRINT PAPE
R 6;"############################
######
   ##        FASES  LUNARES
   ##
   ############################
#####"
  20 PRINT AT 15,6; BRIGHT 1; FL
ASH 1;" AGUARDE UM MOMENTO "
  30 LET G=192: LET H=69: LET J=
58: LET K=52: LET M=108: LET Z=0
  40 GO SUB 510: GO TO 370: BORD
ER 5: INK 0: PAPER 5
  50 FOR D=0 TO 28
  60 CLS : INK 2: PRINT #0;AT 0,
8; PAPER 6;"LUZ SOLAR"; FLASH 1;
 INVERSE 1;" ↑ ↑ ↑ ↑ ↑ ↑ ↑ ": FL
ASH 0: INVERSE 0: INK 0
  70 GO SUB 480: CIRCLE G,H,J: C
IRCLE G,H,12: REM TERRA E ORBITA
 LUNAR
  80 LET P=PI: LET A=(D-7)/14*P
```

```
  90 LET C=G+J*COS A: LET E=H+J*
SIN A: CIRCLE C,E,5: REM LUA
 100 FOR N=.5 TO 5: PLOT C+N,E:
DRAW 0,K-10*N: PLOT C-N,E: DRAW
0,K-10*N: NEXT N: REM SOMBRA DA
LUA
 110 FOR N=.5 TO 12: PLOT G+N,H:
 DRAW 0,M-9*N: PLOT G-N,H: DRAW
0,M-9*N: NEXT N: POKE (K*(M*5-92
))+1,(J+11): REM SOMBRA DA TERRA
 120 PRINT AT 0,0;"FASE - DIA ";
D
 130 PRINT BRIGHT 1;("LUA NOVA"
AND (D=0 OR D=28))+("QUARTO CRES
CENTE" AND D=7)+("QUARTO MINGUAN
TE" AND D=21)+("LUA CHEIA" AND D
=14)+("crescente" AND (D>0 AND D
<7 OR D>7 AND D<14))+("minguante
" AND (D>14 AND D<21 OR D>21 AND
 D<28))
 140 BEEP .2,5: PRINT BRIGHT 1;
FLASH 1;("POSSIVEL ECLIPSE SOLAR
" AND (D=0 OR D=28))+("POSSIVEL
ECLIPSE LUNAR" AND D=14)
 150 PRINT ("CAMINHO DO AMANHECE
R" AND D<14)+("CAMINHO DO OCASO"
 AND D>14)
 160 PRINT AT 11,22;"TERRA";AT 5
,18;"ORBITA LUNAR"
 170 IF D>14 THEN LET P=-P
 180 PLOT J,10: DRAW 0,J*2,P: RE
M ARCO DIREITO DA LUA
 190 LET G=192: LET H=69: LET J=
58: LET K=52: LET M=108: LET Z=0
 200 IF D>14 THEN POKE (K*(M*4+1
6)),116-1: GO TO 230
 210 IF D=14 THEN POKE 23296,(58
-(4*D+2)): GO TO 230
 220 POKE (K*(M*4+16)),(58-(4*D+
3))
 230 LET B=D-7: LET X=2.5
 240 IF B>7 THEN LET B=B-14
 250 LET N=X*ATN (PI/180*-B*25)
 260 PLOT J,10: DRAW 0,J*2,N: RE
M ARCO DIREITO DA LUA
 270 PLOT J,10: DRAW 0,J*2,-P: R
EM ARCO ESQUERDO DA LUA
 280 RANDOMIZE USR 31955
 290 IF Z=1 THEN GO TO 350
 300 PAUSE 200: NEXT D: LET Z=1
 310 BEEP .2,5: PRINT #0; BRIGHT
 1;"  QUER GRAVAR A IMAGEM ? (S/
N)  ": PAUSE 0
 320 LET S$=INKEY$
 330 IF S$="S" OR S$="s" THEN SA
VE *"FASE.SCR"SCREEN$
 340 CLS : GO TO 370
 350 BEEP .2,5: CLS : INPUT " ES
COLHA O DIA DO CICLO (0 A 28) ";
D: LET D=D: FOR D=D TO D
 360 GO TO (D>=0 AND D<=28)*60
 370 BEEP .2,5: CLS : PRINT PAPE
R 6;"###########################
######
     ##        FASES  LUNARES
     ##
     ###########################
#####"
 380 PLOT 0,135: DRAW 0,-130: DR
AW 255,0: DRAW 0,130
 390 PRINT AT 9,2;"1 - VER 28 FA
SES DA LUA "
 400 PRINT AT 12,2;"2 - VER UMA
DAS FASES DA LUA"
 410 PRINT AT 15,2;"3 - TERMINAR
": PAUSE 0
 420 LET K$=INKEY$
 430 IF K$="1" THEN GO TO 50
 440 IF K$="2" THEN CLS : GO TO
350
 450 IF K$="3" THEN STOP
 460 GO TO 420
 470 REM COR DO FUNDO
 480 FOR I=20 TO 0 STEP -1: PRIN
T AT I,16; PAPER 6;"
   ": NEXT I
 490 RETURN
 500 REM ROTINA DO SOMBREADO EM
CODIGO MAQUINA
 510 RESTORE
 520 FOR N=31955 TO 31955+262
 530 READ A: POKE N,A: NEXT N
 540 DATA 42,0,91,124,254,175,20
8,205,143,125,166,254,0,192,1,25
5,255,197,42,0,91,205,143,125,16
6,254,0,32,9,42,0,91,44,34,0,91,
32,236,17,0,0,42,0,91,45,34,0,91
,42,0,91,229,205,143,125,182,119
,225
 550 DATA 124,254,175,40,44,123,
254,0,32,16,36,205,143,125,166,2
54,0,32,7,42,0,91,36,229,30,1,42
,0,91,123,254,1,32,15,36,205,143
,125,166,254,0,40,6,30,0,24,2,24
,167,42,0,91,124,254,0,40,40,122
,254,0,32,16,37,205,143,125,166,
254,0,32,7,42,0,91,37,229,22,1,1
22,254,1,32,14
 560 DATA 42,0,91,37,205,143,125
,166,254,0,40,2,22,0,42,0,91,125
,254,0,40,12,45,34,0,91,205,143,
125,166,254,0,40,129,225,34,0,91
,62,255,188,32,177,189,32,174,20
1,197,213,62,175,148,103,229,230
,7,198,64,79,124,203,31,203,31,2
03,31,230,31,71,230,24,87,124,23
0,192,95,97
 570 DATA 125,203,31,203,31,203,
31,230,31,111,123,128,146,95,22,
0,229,213,225,41,41,41,41,41,209
,25,209,123,230,7,71,62,8,144,71
,62,1,135,16,253,203,31,209,193,
201
 580 RETURN
 590 STOP
9990 SAVE "FASES LUA" LINE 1
```

```
FASE - DIA 7
QUARTO CRESCENTE

CAMINHO DO AMANHECER
```

NOVOS PROGRAMAS

GRÁFICOS II — Compatível com Floppy Disk

Permite obter gráficos com projecção dos dados em áreas de até 24 itens no eixo xx e projectá-los tridimensionalmente. Possui gráficos de linhas e permite projectar duas áreas. No painel principal terá ao seu dispor 14 opções que lhe iriam permitir uma série de acções.

THE BULGE/Lothlorien — Argus

Este jogo é baseado na batalha de Antuérpia que aconteceu nos finais de WWII, este programa é bastante parecido com os Stonkers, usa o cursor para mover as tropas à volta do écran, da mesma maneira. O jogo permite dois jogadores e é um óptimo jogo de estratégia.

VÍDEO POOL/OC

No écran aparece-lhe uma mesa de POOL com 6 cavidades, que poderá alterar para mais pequenas ou maiores.
O jogo oferece três variações de POOL, que dizem respeito ao aumento de dificuldade. Para principiantes só existem 6 bolas, com os números correspondentes nas cavidades. Este jogo tem algumas diferenças, do jogo na realidade, no entanto vale a pena.

BODY WORKS/GENESIS PRODUCTIONS

Do mesmo autor das séries televisivas, o corpo humano, que ultimamente tem sido entre nós apresentado, aparece agora BODY WORKS, um programa que mostra clara e ilustradamente em detalhe as várias funções do corpo. As células, respiração, digestão, músculos, nervos, circulação, etc., são claramente reproduzidas.
Se quer saber como uma mensagem é transmitida dos nervos ao cérebro, introduza este programa e vá aprendendo. Cada função está ilustrada graficamente. Tudo o que tem a fazer é accionar SPACE, quando quiser passar para a página seguinte.

STARION/MELBOURNE HOUSE

Este programa é constituído por uma série de puzzles com o bom estilo gráfico combinado com os efeitos estratégicos a três dimensões. Trata-se de uma retrospectiva histórica, com 243 acontecimentos ocorridos nos últimos 100 anos na história da terra. O jogo está dividido em grelhas de tempo e zonas agrupadas em blocos de nove. Em cada zona tem que se apanhar letras para formar uma palavra que o levará ao tempo desejado.

DEATH STAR/IMAGINE

Neste jogo começará por fazer a descolagem, quando ouvir o Spectrum emitir um som a nave será lançada e terá que guiá-la por uma janela até ao espaço exterior, verá a terra a afastar-se e a DEATH STAR a aproximar-se da próxima secção, no quadro seguinte aparecem naves que terão de ser evitadas ou destruídas, por fim terá que colocar uma bomba.

SPY HUNTER/US GOLD

Este é um dos jogos de arcadia com maior sucesso. Neste jogo considera-se a fuga de um espião através de um país, com uma série de agentes na sua pista.
Felizmente o seu carro foi construído para James Bond, ou seja pode andar na água sem problema de afundar.
A caça é rápida violenta e o carro faz muitas derrapagens, pode atirar contra os agentes, tentando tirá-los da estrada, mas se apanhar em outra pessoa que não tenha nada a ver com o assunto, perde pontos. Tem 2 fases de dificuldade, uma para participantes e outra para experts.

LIVROS PARA VENDA EM FOTOCÓPIAS

Título	Preço
Introducing Spectrum Machine Code	350$00
Basic And Games, Computers and Cheesecake	350$00
La pratique du ZX Spectrum	450$00
La conduite du ZX Spectrum	450$00
Starting Forth	500$00
40 Best Machine code Routines for the ZX Spectrum	350$00
La Pratique du ZX Spectrum et du Timex 2000	300$00
Jeux et Aplications pour ZX Spectrum	300$00
Basic Hidraulics	370$00
Spectrum Microdrive Book	300$00
Machine Language Simple Sinclair	350$00
Games for Your ZX Spectrum	200$00
The Working Spectrum	450$00
Over The Spectrum	300$00
ZX Spectrum Teoria y Projectos de Interfases	400$00
The Spectrum Programer	350$00
Mastering Machine Code on Your ZX81	200$00
Spectrum Machine Language for the Absolute Beginner	300$00
Games to Play on Your Zx Spectrum	100$00
Better Programming for your Spectrum and ZX81	450$00
The Spectrum Handbook	450$00
Sinclair ZX Spectrum	450$00
Dynamic Games for the ZX Spectrum	450$00
The Hobbit	350$00
20 Simple Electronics Projects for ZX81 and Spectrum	350$00
Jupiter Ace	350$00
The Explorer Guide to the Zx81	150$00
Sinclair ZX81 Rom Disassembly	100$00

LIVROS PARA VENDA A PREÇOS ESPECIAIS

Título	Preço
The Spectrum Pocket Book	350$00
the ZX Spectrum and How to Get the Most From It	300$00
Spectrum Microdrive Book	300$00
Spectrum Spectacular	350$00
Spectrum Graphics	350$00
The Spectrum Games Companion	250$00
Educacional Uses of the ZX Spectrum	350$00
Games for Your ZX Spectrum	200$00
Over the Spectrum	200$00
Spectrum Interfacing and project	300$00
ZX81 Basic Programing (em Portugues)	50$00
60 Games and Applications for the ZX Spectrum	150$00
The Spectrum Explored	350$00
Spectrum Machine Code Made Easy vol 1	400$00
Spectrum Machine Code Made Easy Vol 2	400$00
Exploring Spectrum Basic	400$00
The Working Spectrum	450$00
Machine Language Simple Sinclair	350$00
Programing Your Zx Spectrum	400$00
Sixty Programs for the Sinclair ZX Spectrum	400$00
Spectrum Adventures	400$00
The ZX81 Pocket Book	70$00
Master Your Zx Microdrive	300$00
The Spectrum Book of Games	300$00
Understanding Your ZX81 ROM	100$00
Spectrum Hardware Manual	350$00
ZX Spectrum Microdrive and Interface 1 Manual (port. e Ing.)	200$00
Easy Programing for the ZX Spectrum	300$00
ABC dos Computadores	300$00
Manual do ZX Spectrum	450$00
O Computador no Escritório	400$00
ABC da Programação de Computadores	300$00
Guia do Sinclair QL	420$00
Introdução a Programação de Microcomputadores	400$00
Novas Aventuras no seu ZX Spectrum	450$00
26 Programas Basic	430$00
Código Máquina para Programadores Avançados	430$00
Z80 Assembler para o ZX Spectrum	400$00
Understanding Your Spectrum	400$00
The Complete Spectrum ROM Disassembly	400$00
Clefs Pour Le ZX Spectrum et Le Timex 2000	200$00
Advanced Graphics With the Sinclair ZX Spectrum	500$00
The Micro User's Book of Tape Recording	350$00
20 Best Programs for the ZX Spectrum	300$00
Jogos e Programas em Basic	400$00

ANÚNCIO

O nosso associado João Florêncio Bárbara Nunes pede que o ajudemos no jogo de aventura Valkyrie 17, pois não consegue que alguma coisa aconteça, além de vaguear pelas várias dependências do hotel e encontrara num cofre uma jóia e um gás venenoso.

Pede também que lhe dêem umas pistas no jogo Sherlock, que traduzam a mensagem em código que se encontra na casa de Basil.

Se algum associado, quiser ajudar este nosso amigo, agradecemos que nos enviem as respostas.